Procedia Computer Science

Volume 51, 2015, Pages 650–659

ICCS 2015 International Conference On Computational Science

Towards an Integrated Conceptual Design Evaluation of Mechatronic Systems: The SysDICE Approach

Mohammad Chami[1,2] and Jean-Michel Bruel[2]

[1] Bombardier Transportation GmbH
mohammad.chami@de.transport.bombardier.com
[2] IRIT/University of Toulouse
bruel@irit.fr

Abstract
Mechatronic systems play a significant role in different types of industry, especially in transportation, aerospace, automotive and manufacturing. Although their multidisciplinary nature provides enormous functionalities, it is still one of the substantial challenges which frequently impede their design process. Notably, the conceptual design phase aggregates various engineering disciplines, project and business management fields, where different methods, modeling languages and software tools are applied. Therefore, an integrated environment is required to intimately engage the different domains together. This paper outlines a model-based research approach for an integrated conceptual design evaluation of mechatronic systems using SysML. Particularly, the state of the art is highlighted, most important challenges, remaining problems in this field and a novel solution is proposed, named SysDICE, combining model based system engineering and artificial intelligence techniques to support for achieving efficient design.

Keywords: SysML, System Modeling, Mechatronics Design, Design Evaluation, SysDICE

1 Introduction

Mechatronics engineering, with its *"synergetic integration of mechanical engineering, electrical engineering and computer science"* [24], has been considered as one of the main innovation leader in industry. Namely, it provides new prospects for higher level of innovation, higher performance products and a wide range of functionalities. Traditionally, the design and development process of mechatronic systems iterates over three phases: synthesis, analysis and evaluation [23]. In each of these phases, a wide range of languages, methods, and tools are used. Particularly, the conceptual design phase is the part of the design process where a *"solution principle"* is specified and here with *"evaluation"* it is meant to determine the value, usefulness or strength of a solution with respect to a given objective [16]. System engineers play a crucial role in performing such an evaluation as they hold the knowledge base of all involved domains (from requirements, down into functions and high-level design solutions) and their dependencies.

 Selection and peer-review under responsibility of the Scientific Programme Committee of ICCS 2015

doi:10.1016/j.procs.2015.05.180

While investigating within the problem space of mechatronics, one frequently comes across the integration issue. The inner part of the V-model in Figure 1, shows typical mechatronic design entities. It is necessary to consider in details a set of these entities in order to realize a particular function the product should perform. For instance, upon a change of a requirement, e.g., "increasing the maximum speed of a car from 180Km/h to 220Km/h", one have to start analyzing from this requirement, moving forward to the refined functions, concepts and proposed solutions while considering all discipline specific criteria. Consequently, the complexity arises due to the fact that this entities level is directly dependent on the other factors such as the tools, methods and human factors levels.

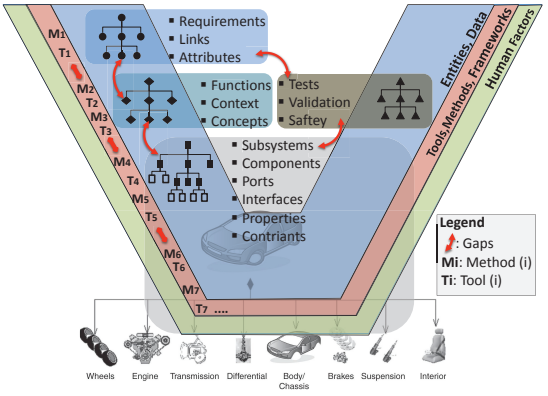

Figure 1: Part of the conceptual design entities context according to the V-model [6, 24]

From one side, the integration problem is not only concerned with the entities and data level. However, it involves the tools, methods, processes and frameworks used among the human factors. In reality, system engineers still follow a document-based manner to hold the disciplines' interdependencies (i.e., how, when and in what way any discipline influences another). This frequently leads to weak synchronization and result in inefficiencies that often appear only during the integration or testing [6]. In order to avoid these problems, an integrated environment is demanded to involve all the various expertise. Hereby, the usage of models and model transformations to bridge such integration gaps and the initiatives of Model-based System Engineering (MBSE) are the most actual promising approaches in this field.

From another side, the issue of complexity is getting more and more crucial. Nowadays, alone the car entertainment system, brings up more than hundred thousands of requirements and high level of safety issues which are necessary for finishing a homologation process. Typical system models possess a complicated structure with various levels of abstraction and wide range of view points. Additionally, it is not only about involving many components but also considering their influence on each other [16]. Therefore, additional mathematical formulation and computer support is definitely required. Hereby, adopting suitable Artificial Intelligence (AI) techniques for such models have proven to give rise to decreasing model execution computation time [2].

Until now, little attention has been given to collaborative work for evaluating designs [26]. Therefore, the topic of evaluating integrated designs still needs to be deeply studied. This paper explores an integrated conceptual design evaluation approach for mechatronic systems based on SysML, which we have called SysDICE. Notably, The Systems Modeling Language (SysML) [8] is used as the common language between the discipline engineers (with their concrete discipline-information about the system) and the system engineers (with their view of the system as a whole) to form a *system model*. Later on, this system model is formalized and transferred accordingly in order to make it executable and applicable for AI algorithms aiming at evaluating design criteria during the conceptual design phase.

The rest of the paper is organized as follows: Section 2 presents a brief background and describe the state of the art. After the research objective and SysDICE approach are described in Section 3, we present in Section 4 the actual status of the approach with an application example. Finally, Section 5 is dedicated for concluding the paper and giving an outlook.

2 Background and State of the Art

2.1 Methodologies, Processes, Frameworks and Tools

From mechatronics engineering perspective, the *methodology* is defined as the way how products are designed, developed and produced. Tomiyama et al. [23] present an excellent description of the *design theory and methodology* (DTM) and an evaluation of its application in practice. Obviously, several methodologies have been developed, with a lot in common, as for Pahl and Beitz [16] and the VDI2206 [24]. Nevertheless, one have to accept the fact that from literature side, it is agreed that there is no *"one accepted methodology"* [23] and from other practical side, this problem seams to be hardly solved as companies are individually developing their own methodologies. Moreover, it is definitely crucial to take into account the methodologies' usage in practice, focus on their evaluation and consider the goals behind applying them.

From system engineering perspective, a *process* defines what activities are performed and does not generally give details on how they are done [6]. Several process approaches have evolved within the system engineering (as the Traditional, Top-Down Systems Engineering (TTDSE) process [19]), to other standards as IEEE1220 and ISO15288. Moreover, software engineers have also evolved several approaches, from waterfall process, to spiral development, and more recently to the object oriented design. Although, these approaches have solved some of the organizational and technical problems, they are also rarely applied in the big industries.

In addition to mechatronic methodologies and system engineering processes, several frameworks have matured to apply them. For instance, the Department of Defense Architecture Framework (DoDAF) [5], supports the defense industry by defining the architecture's operation, system and technical views, but it is still considered complicated and extensive for specific systems. Whereas, the Model Driven Architecture® (MDA®) [14] aims at deploying, maintaining and integrating with lower costs by using models in software development. Hereby, engineering tools are a key factor in forming such productive frameworks. Generally speaking, tools used during the design process can be categorized into three types:
(1) **Domain-Specific Tools** (DST), for instance mechanical engineers employ different CAD tools for their engineering drawings and analysis, which is the similar case for electrical and control engineers for simulation, whereas software engineers still focus during the design process more on code rather then modeling.
(2) **Domain-Coupling Tools** (DCT), such as MATLAB/Simulink, Simscape, Modelica, are used intensively in industry. These tools have been popular by stepping one way towards the system level and involving more then one discipline during the development and simulation.
(3) **One-Tool Concept** (OTC), which consider large heterogeneous systems, exist on the market, e.g., Mechatronics Concept Designer and Dassault Systèmes Enovia®. Such tools support integration but require a multidisciplinary knowledge about the system and they still can be hardly competitive with the DSTs.

Although the DSTs are the most popular, their integration remains extremely challenging and they are often used beyond their scope of applicability. Additionally, problems still lie ahead while dealing with complexity, variant management, and tools' updates. Therefore, it is well agreed that DSTs should not be used on a high-level and for multidisciplinary systems. Instead the system modeling tools should be applied. Thus, we see a great benefit by integrating both DSTs and DCTs with the system modeling tools rather then providing a new OTC.

Tools integration problems, seams to be solved with the new promising open community, the *Open Service for Lifecycle Collaboration* (OSLC) [15], which is developed for enabling the integration of software development and more broadly Application Lifecycle Management and Product lifecycle Management products. However, it is still in its early stages of development.

2.2 Modeling and The Common Language Challenge

One of the challenges of mechatronics design is a successful integrated modeling method, where a common language to model the different disciplines is required [9]. Although, various domain-independent modeling methods have been used, for instance, the *bond graph, Petri nets, N-squared charts and Finite-state machine*. A formal representation of the common information combining these methods is still missing. Therefore, this common representation is still often specified in a document-based manner and hardly mapped to the actual models' information.

The common language challenge have been the topic of several research approaches since decades. Many researches have followed a component based approach to represent the elements of a mechatronics system for their own needs as in [4, 22]. Zhang et al. [26] developed their own multi-view modeling paradigm to support the collaboration work of designers. Chen et al. [4] propose a constraint modeling-based approach by modeling the components of mechatronic systems as objects with attributes, and by identifying and modeling the constraints between these attributes. Unfortunately, these approaches highlight one piece of the puzzle, the "modeling language" piece. While applying them to different type of systems and with different modeling goals, other puzzle pieces, i.e, the "method" and the "tool" limits their application.

Others have followed a UML-based modeling for the mechatronic design, such as the Mechatronic UML [20] which allows a model-driven development while supporting verification and code generation. Hereby, SysML came after UML to solve some limitations for system engineering applications. In the following, we highlight the SysML related work and its execution.

2.2.1 SysML Related Work

Although SysML is only few years old, a wide range of researchers and industries have applied it for their different needs. SysML-based information models have been proven to be useful for formal information and knowledge capturing. As previously mentioned, a generalized common language for modeling the multidisciplinary information in mechatronics design is still missing. Generally, SysML with its diagrams deals with this problem and has been already successfully adopted during the last few years for modeling mechatronic systems as in [3, 17, 21, 22]. Namely, in [3, 17] the system-level modeling with SysML was adopted to support mechatronic design. In [21] SysML profiles were particularly applied to support the multi-view modeling approach and in [22] SysML was used to specify the central view-model of the mechatronics system.

From a requirements engineering point of view, various methods dealing with requirements analysis and traceability have been proposed. However, the linking between requirements and other model entities (i.e., components, properties) is hardly documented. Although SysML supports in requirements modeling and consider particularly this linking, the industrial usage of SysML for requirements analysis and requirements engineering is still not so mature. Commonly, requirements are imported to SysML tools in order to be linked to other SysML elements. This importing mechanism is still inefficient and requires high maintenance effort. Hereby, OSLC [15] solves this problem however, it is still in early development phase and not yet applied in productive industrial applications.

2.2.2 AI applications for System Models' Formalization and Execution

AI methods have been proposed to aid the mechatronic design process. For instance, in [13] the design activity optimization was solved using a heuristic-based hybrid search algorithm and in [25] a maximum likelihood estimation method for determining the unknown design parameters based on given information was conducted. The application of ant colony optimization

(ACO) for combinational optimization and particle swarm optimization (PSO) for continuous optimization is described in [1]. An efficient swarm intelligence (SI) based algorithm for multi-objective optimization is presented in [18] where the corporation of a Pareto dominance relation into PSO was proposed. It is generally agreed that the main problem in these existing approaches relates to the high effort in capturing the interdisciplinary information to be used in AI. Although others [11], proposed an integrated design evaluation, with graph based models and usage of PSO for encoding such models, they are considered as non-generalizable due to the limitations of the graph based modeling approach.

The formalization of SysML models has been also considered. For instance, Petri nets and temporal logic LTL are used in [12] to formalize the system behavior and requirements, and in [7] some SysML diagrams are encoded with description logic for formal semantics. Compared to these approaches we aim to take a step further in incorporating noisy models (i.e., models which don't exist in reality) and uncertainties (having a configurable error range) that are typically not available once adopting logical descriptions. Actually, we use a Gaussian noise to allow the values of requirements and properties to be uncertain. This mechanism tends to generate noisy-models with bigger solution-space. These models are later used as input knowledge for a particular evaluation objective(s) in order to find the most suitable solution (real-model).

3 Research Objective and SysDICE Approach

Our research scope concerns mainly the usability of MBSE approaches and AI techniques for supporting the mechatronic design. This scope environment is the result of previous investigations and published work. Starting from [3], a SysML-based integration framework was proposed to bring the different disciplines together for a better collaboration. Particularly, different general purpose modeling languages have been analyzed and SysML have been seen to be the most promising approach for this manner. Moreover, to achieve the collaboration, SysML model elements were transfered into a multi-agents system and mapped to other agents from the process model elements. Afterwords, the scope was extended towards adopting AI techniques for executing the SysML model while supporting the system design evaluation [2].

In Summary, an early integrated evaluation of the system design, as a whole, in a sequel of making the procedure adaptable, efficient and intelligent is what this research work aim to pursue. Notably, a SysML-based method is proposed for an Integrated Conceptual Design Evaluation of mechatronic systems, abbreviated as SysDICE. This aims at attaining an efficient system design process and thus leading for short time and cost effective mechatronic products. In the following, the overall framework and methodology of SysDICE is described. Notice that the tool implementation is still in its early stages and it is outside the scope of this paper.

SysDICE Overall Framework and Methodology. Figure 2 presents a high level scheme of the proposed framework. We categorize the human factors involved into (1) Discipline and (2) System engineers. For the first group, a discipline-specific information can be represented in SysML while assuring that the SysML details level is restricted to only the amount of information needed for achieving a cross-discipline mapping. For the second category, system engineers, can model system requirements, functions, the abstract conceptual solution (i.e., structure, behavior and constraints) and manage the system model using SysML. They are able to evaluate the system design model through the tool solver which is running in the background to provide the execution of the SysML model. Furthuremore, the top part of Figure 2 shows three main steps of SysDICE general methodology:

Step 1: The system model generation, where a SysML tool is used with the support of the SysDICE profile for forming the system model. SysDICE profiles are used to extend the SysML metamodel for modeling the domain specific aspects. On the one hand, they should support in validating the activities' outcomes and on the other hand they handle the identification of model elements and mapping it to discipline tools' elements via the tool adapters. Figure 2 indicates further six types of modeling activities (evaluation, requirements, functional, structure, behavior and constraints). Each of these activities results in a set of SysML elements and relations shown with the respective SysML diagrams. These from the multidisciplinary system model, which we split here into three levels:

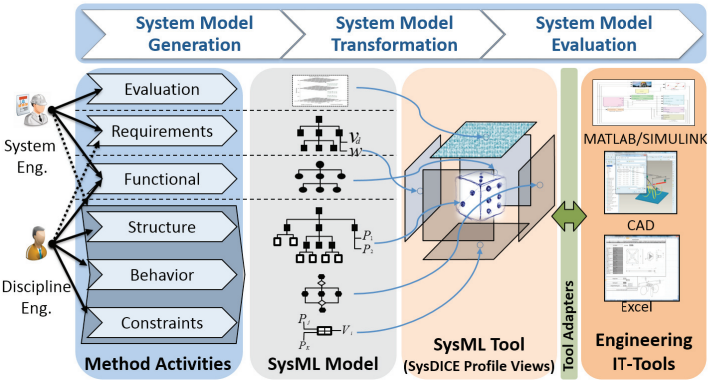

Figure 2: SysDICE overall framework

1. The system's requirements which are classified as (a) numerical requirements with their desired numerical values and weighted priorities (e.g., total weight of 2 Kg with 70% priority) and (b) non-numerical requirements with their desired textual description and weighted priorities (e.g., lowest possible response time with 90% priority).
2. The system's functions which refine and describe the non-numerical requirements more in details and clarify its text based information with functions indicating what the user expects from the product, and
3. the system's conceptual design solution which includes (a) the hierarchy of the components together with their respective parameters and behavior (i.e., components here can be interdisciplinary, mechatronics, such as a motor with motor board controller or discipline-specific; chassis as mechanical, electronic board as electrical or pure software code) and (b) the interrelationships between disciplines through the constraints with their corresponding input and output properties (e.g., power consumption, operational time, total price).

Step 2: The system model transformation, which implicitly includes the mathematical formulation of the system model and assures transferring it to an executable version. Actually, the generated model is parsed and converted into a mathematical solver (i.e. the actual used mathematical solver tool is MATLAB) for evaluating different model configurations performed by system engineers. Hereby, consistency and model validation are major parts which reports about the quality of the generated model before performing the evaluation step.

Step 3: The system model evaluation, which involves the evaluation activity (seen in Figure 2) starts with capturing and identifying the design evaluation criteria (by stereotyping the respective requirements as evaluation goals) and ends with providing them to the transferred model for applying the mathematical algorithms. The evaluation results represent the feedback-loop for optimizing the conceptual solution upon particular evaluation goals' configuration.

Certainly, the three steps of of SysDICE general methodology are performed in an iterative and evolutionary manner until the system engineer come to the required optimum solution. In the following section we demonstrate this with an application example.

4 Preliminary Work and Application Example

The design of a two wheel differential drive robot illustrates the application SysDICE for modeling the robot with SysML (via MagicDraw tool) and applying the mathematical formulation to find the optimal combination of components alternatives for a specific evaluation goals. This is described in the following three fundamental steps of SysDICE:

Step 1: Generate the SysML robot model. During early design stages a set of requirements spanned over the various domains is provided. With SysDICE, each of these requirements is modeled using the ≪*requirement*≫ block within the req diagram (Figure 3(1)). To be fully able to specify a numerical design requirement, we extend the existing SysML requirement by stereotyping it to include its "value", v_d and its corresponding "priority", w, (shown on the "*Total Weight*" requirement). We call this stereotype, ≪*EvaluationGoal*≫ as it represents later for the optimization engine the evaluation objectives source information. We further identify a non-numerical requirement to indicate the necessity of associating it to its respective function with the ≪*refine*≫ association. Regarding the functional modeling, SysML doesn't offer a particular functional diagram but it offers the *use case* diagram instead where a highest level of abstraction is represented for the interaction between the system and its external actors [10]. The use case diagram have been used in [6, 10] to refine the functional requirements. Hereby, this method is adopted for representing each of these functions using the ≪*usecase*≫ element and further represent their hierarchy using the ≪*include*≫ association as shown in Figure 3(2).

After the design requirements have been settled, system engineers commence to generate a conceptual solution. At this stage, the system evolves from a black box to detailed subsystems reaching the component levels. Following a similar trend, our framework then decomposes the robot into its constituent subsystems and their corresponding components. This is achieved through the SysML ≪*block*≫ element, which is stereotyped as ≪*component*≫, and the ≪*composition*≫ association within the bdd diagram. Each component of the robot could have various alternatives which are stereotyped as ≪*AlternativeComponent*≫, in order to represent their uniqueness in a possible design solution, and related to the respective component with the ≪*Variant*≫ generalization relation (Figure 3(3)). Moreover, they are specified by their corresponding properties (such as weight, price, power consumption). The relations between these properties are modeled using the ≪*constraintProperty*≫ within the par diagram (Figure 3(5)), and the interfaces between the components are modeled within the ibds (Figure 3(4)).

Additionally, Figure 3(6) shows the ≪*satisfy*≫ and ≪*refine*≫ relationships matrix between the properties and use cases respectively towards the requirements. At this stage a SysML model, which incorporates all the disciplines, is generated. Therefore, the necessary information for system engineers is ready for evaluation and the integration burden is solved.

Step 2: Formulate and transfer the SysML robot model. The mathematical formalization of the weighted requirement satisfaction problem with the multi-alternative mechanism is divided into two levels of abstraction:

Abstraction Level One: Given a set of k requirements, $\mathbf{v}_d = [v_d^{(1)}, \ldots, v_d^{(k)}]^T \in \mathbb{R}^{k\times 1}$ is defined to represent the different desired values of each of the numerical requirements, and $\mathbf{W}_{k,k} = diag(\mathbf{w})$ to be the diagonal matrix representing the priorities of each of these requirements. We further define $\mathbf{v} = [v_1, \ldots, v_k]$, to represent the output of the *constraintProperty* equations which relate a set of properties as its inputs.

It is assumed that these values are uncertain (having a configurable error range), noisy with a Gaussian noise, and that the requirements are weighted in each of the k directions according to their priorities. Therefore, the likelihood for a desired value to occur is defined by:

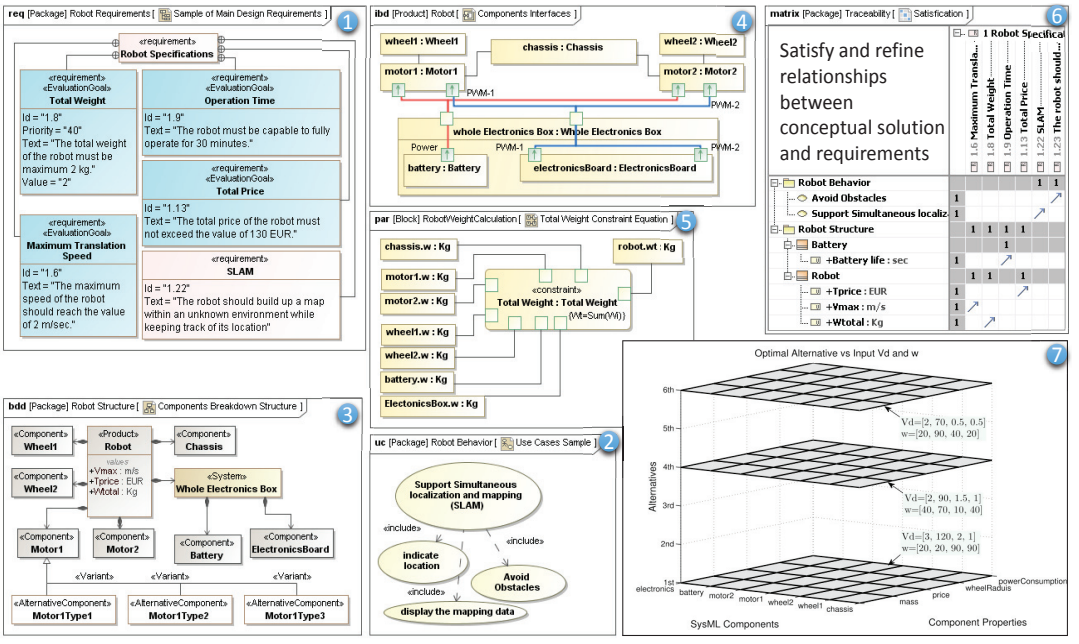

Figure 3: SysML Diagrams and results of three different evaluation goals configurations

$$p(v_d^{(i)}|v^{(i)};\sigma^2,w^{(i)}) = \prod_{i=1}^{k} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2\sigma^2} w_{i,i}(v_d^{(i)} - v^{(i)})^2\right) \quad (1)$$

Maximizing the natural logarithm of Equation 1, leads to the following minimization problem,

$$\min_{\mathbf{v}} \frac{1}{2}[\mathbf{v} - \mathbf{v}_d]^T \mathbf{W}[\mathbf{v} - \mathbf{v}_d] \quad (2)$$

Equation 2 represents the weighted requirement satisfaction problem. In other words, the solution of the minimization problem is seeking the optimal value, $\mathbf{v}^*$, so as to minimize the error with respect to the desired combination weighted by the priorities (i.e., $\mathbf{v}^* = \arg\min_{\mathbf{v}} \frac{1}{2}[\mathbf{v} - \mathbf{v}_d]^T \mathbf{W}[\mathbf{v} - \mathbf{v}_d]$).

Abstraction Level Two: To approximate the values of the corresponding combination of the properties, we resort to Gaussian Processes (GPs). The reasons for our choice are threefold: (1) the constraint equations are complex and thus require non-parametric functional approximators, (2) the lack of available training data which imposes good generalization properties of the used approximators, and (3) the need for a problem independent framework.

The approximated functions are then substituted in Equation 2, to generate a new minimization problem defined by the following cost function,

$$\min_{P} J(P) = \frac{1}{2} \sum_{i=1}^{k} w_{i,i}(\mathcal{GP}_i(P) - v_d^{(k)})^2, \quad (3)$$

where $P = p_1 \bigotimes p_2 \cdots \bigotimes p_N$, with N being the number of components, and J representing the cost function. To minimize Equation 3, we need to compute the derivatives with respect to the input. Here we approximate the derivate of a GP using first order approximation and then use conjugate gradient descent for the optimization. The output is P^* that satisfies the set combination of the prioritized requirements (i.e., $\arg\min_P \sum_{i=1}^{k} w_{i,i}(\mathcal{GP}_i(P) - v_d^{(k)})^2$).

Step 3: Evaluate the best components combination of the SysML robot model. To better evaluate the framework, we have conducted various experiments with different priorities and desired requirements' values. Moreover, the system was provided with different alternatives having various properties and the model is parsed in order to provide the required information for the algorithm. After the GPs were approximated, conjugate gradient descent was applied to find the optimal alternative suiting the requirements. Figure 3(7) shows the results from MATLAB providing the different values and priorities. The three axis of the graph represent the components, properties and the alternatives respectively. The different planes are the optimal alternatives resulting from different requirements values and priorities. Each of these priorities and/or properties change represents a different design focus. For instance, in the middle plane the focus was more towards having a relatively medium price (i.e., 90), where the total price was given a priority of 70%. The upper one correspond to a focus towards having a cheap price of 70 with a high priority (i.e., 90%). It becomes obvious from Figure 3(7) that the platform captures different optimal alternatives suiting different design focuses and requirements and thus being adaptable and generalizable to different requirement and or priority values.

5 Conclusion

In this paper, we explored the application of MBSE and AI for supporting the mechatronics design. SysDICE, contributed by making use of SysML as a common language between system and discipline engineers. Furthermore, it was capable of representing the interdisciplinary interrelations that usually complicate the design process. SysDICE method was described with an application example. The model generation phase showed how SysML diagrams were used to model the requirements, functions, and conceptual solution entities. The method further made use of Gaussian Processes in order to find a functional mapping at the system-design level. These were then used to solve for the best alternative combination that optimally suits a set of configured requirements. Experiments conducted on the design of the robot show the accessibility and adaptability of the approach, whereby the framework was capable of bridging the system engineering level communication problems, attaining optimal alternatives to a set of requirements, and producing adaptable solutions to various design focuses.

There are a lot of interesting directions for future work. Here we aim to extend the solution space and clarify concrete steps regarding the method and tool development. In this paper, we solved the weighted requirement satisfaction problem only for the numerical requirements and thus our next goal is to cover also the non-numerical requirements. On a higher level, the actual system model will be divided into a generic and project specific parts. It is aimed to support reusability and knowledge sharing via using the generic part in different projects. Thus we would be able also to perform design evaluations of of Systems of Systems design models.

References

[1] Christian Blum and Xiaodong Li. *Swarm Intelligence in Optimization*. Springer, Natural Computing Series, Swarm Intelligence, Part I, 2008.

[2] Mohammad Chami, Haitham Bou Ammar, Holger Voos, Karl Tuyls, and Gerhard Weiss. Swarm-based evaluation of nonparametric sysml mechatronics system design. IEEE International Conference on Mechatronics, ICM 2013, IEEE, 2013.

[3] Mohammad Chami, Holger Seemller, and Holger Voos. A sysml-based integration framework for the engineering of mechatronic systems. IEEE/ASME International Conference on Mechatronic and Embedded Systems and Applications, IEEE, 2010.

[4] Kenway Chen, Jonathan Bankston, Jitesh H. Panchal, and Dirk Schaefer. *A Framework for Integrated Design of Mechatronic Systems*, chapter 2, pages 37–70. Springer, 2009.

[5] DoDAF. "US Department of Defense Architecture Framework Working Group, Department of Defense Architecture Framework (DoDAF), Version 1.0"., February 2004.

[6] Sanford Friedenthal, Alan Moore, and Rick Steiner. *A Practical Guide to SysML: The Systems Modeling Language*. Elsevier, Morgan Kaufmann OMG Press, 2008.

[7] Henson Graves and Yvonne Bijan. Modeling structure in description logic. *DL2011*, 2011.

[8] Object Management Group. "OMG Systems Modeling Language (OMG SysML^{TM})". available at `http://www.omgsysml.org`, November 2008.

[9] Jon Holt. *UML for System Engineering: watching the wheels*. Institution of Engineering and Technology, second edition, 2004.

[10] Jon Holt and Simon Perry. *SysML for Systems Engineering*. The Institution of Engineering and Technology, London, United Kingdom, 2008.

[11] Feng-Yi Huang and Yuan-Jye Tseng. An integrated design evaluation and assembly sequence planning model using a particle swarm optimization approach. 2011.

[12] Marcos V. Linhares, Romulo S. de Oliveira, Jean-Marie Farines, and Francois Vernadat. Introducing the modeling and verification process in sysml. In *IEEE International Conference. on. Emerging Technologies and Factory Automation (ETFA)*, 2007.

[13] Ouael Mouelhi, Pierre Couturier, and Tanneguy Redarce. An Artificial Intelligence Approach for the Multicriteria Optimization in Mechatronic Products Design. In *Proceedings of the 2009 IEEE International Conference on Mechatronics and Automation*, pages 1731–1736, 2009.

[14] OMG. *Model Driven Architecture (MDA) Guide*, 2003. OMG doc. ab/2003-06-01.

[15] OSLC. "Open Services for Lifecycle Collaboration". available at `http://open-services.net/`.

[16] Gerhard Pahl, Wolfgang Beitz, Jrg Fledhusen, and Karl-Heinrich Grote. *Engineering Design A Systematic Approach*. Springer, third edition edition, 2007.

[17] Ahsan Qamar, Jan Wikander, and Carl During. Designing mechatronic systems: A model-integration approach. In *In Proceedings of the 18th International Conference on Engineering Design (ICED11)*, volume 4, pages 145–156, 2011.

[18] M. Janga Reddy and D. Nagesh Kumar. An efficient multi-objective optimization algorithm based on swarm intelligence for engineering design, 2007.

[19] Andrew P. Sage. *Systems Engineering*. Wiley, New York., 1992.

[20] Wilhelm Schaefer and Heike Wehrheim. Model-driven development with mechatronic uml. In *Graph Transformations and Model-Driven Engineering*, volume 5765 of *Lecture Notes in Computer Science*, pages 533–554. Springer, 2010.

[21] Aditya A.. Shah, Dirk Schaefer, and Christiaan J.J. Paredis. Enabling multi-view modeling with sysml profiles and model transformations. In *International Conference on Product Lifecycle Management*, page 10. Inderscience Enterprises Ltd, 2009.

[22] Kleanthis Thramboulidis. The 3+1 sysml view-model in model integrated mechatronics. *Journal of Software Engineering and Applications (JSEA)*, 3(2):109–118, 2010.

[23] T Tomiyama, P Gu, Y Jin, D Lutters, CH Kind, and F Kimura. Design methodologies: Industrial and educational applications, 2009.

[24] VDI. *VDI 2206 Design methodology for mechatronic systems*. Beuth Verlag GmbH, Düsseldorf, Germany, June 2004.

[25] Xinsheng Xu, Linyun Fu, and Shuiliang Fang. Research on Product Variant Design with Uncertainty Information. In *Proceedings of the 7th World Congress on Intelligent Control and Automation*, Chongqing, China, 2008.

[26] Heming Zhang, Hongwei Wang, David Chen, and Gregory Zacharewicz. A model-driven approach to multidiciplinary collaborative simulation for virtual product development, 2010.